"Method and System for simulating communications networks, object and computer program product therefor"

Field of the invention

The present invention refers to techniques for simulating communications networks such as, for instance, radio-mobile cellular networks.

The simulation is an essential step in planning, designing, realizing and managing such networks, especially in view of the optimisation of performances of the networks themselves. Particularly, the simulation plays an important role both at check level for planning a new network, and at updating and optimising level for performances of an already operating network.

The invention has been devised paying particular care to the techniques of survey of inherent statistic data to the behaviour of the simulated network. Typical examples of statistic data that can be measured in a radio-mobile cellular network system simulator are network capacity, mean delay on packet transfer, percentage of blocked calls, etc.

Description of the prior art

It is known that there are cellular radio-mobile network system simulators characterised by an object architecture, such as disclosed, for example, in WO-A-02/104055. According to the object approach, the elementary decomposition unit is not the operation (method), but the object, meant as model of a real entity (a real-world object).

It is known that in such simulators there are modules or devices adapted to simulate the behaviour of physical network devices. Every module or device can have many implementations (according to simulated

functionalities or technologies) and each of these implementations can be used at will in the simulations: a different way of operating of the module in itself corresponds to each implementation. It is also known that the modules or devices in the simulator mutually exchange information, with the purpose of simulating the behaviour of real physical network devices; the exchange of information occurs through two modes: with "messages" and with "events".

The communication with messages is characterized in that the receipt of information by the target module occurs simultaneously with the dispatch from the source module; instead, the communication with events is characterized in that the receipt of information from the target module does not occur simultaneously with the transmission from the source module, but in a particular following time instant after the transmission time: the use of the events serves for timing the receipt of a piece of information from the receiving module.

It is also known that these simulators are able to perform measurements of the behaviour of the various simulated modules or devices, providing some statistics related to these measurements as simulation results.

In the arrangement described in WO-A-02/104055 the objects-simulator architecture provides an engine in which there is an entity called Statistic Manager. Task of such entity is performing, by intercepting communications between module and module, the measurements related to the behaviour of modules or devices themselves and processing statistics of measured statistic data.

A typical example of statistic processing is the computation of mean transmission delays of a packet. In this case, the Statistic Manager entity performs, for

every packet transmitted in the simulated system, a delay measurement, averaging among them the various measured delays, obtaining the mean delay as result.

It is known that - with analogous operating criteria - all interesting statistic processing in a system simulator can be realised, for instance for radio-mobile cellular networks.

The Applicant has observed that in a situation of the previously described type, various types of problems occur that are intrinsically tied up to the possible and frequent presence of different implementations of the same network functionality.

Firstly, the communication among two modules occurs directly through the respective implementations: in every implementation of a given module, it is therefore necessary to insert, for the Statistic Manager's benefit, all information that identify communication modes with all implementations of the other modules with which the affected module can communicate.

Secondly, the mechanism with which the Statistic Manager measures the behaviour of the modules or devices changes according to implementation, because it has to suit itself for the particular module/device implementation.

Insofar, depending on known prior art, with the purpose to be able to take into account the presence of many implementations of the same device or module, it is necessary to insert in the Statistic Manager a specific interception and measuring mechanism, operation that can be also very costly in terms of time and processing complexity.

It will also be appreciated that this is worth above all for the telecommunications networks (where the same network functionality is performed by

apparatuses supplied by different manufacturers, with different implementations) and that the same type of problems occurs, in substantially identical terms, also in the communication between various modules or devices, therefore in a way that is independent from the intervention of the Statistic Manager or an equivalent entity.

Object and synthesis of the present invention

Object of the present invention is therefore solving the above-mentioned problems.

According to the present invention, such object is reached thanks to a method having the characteristics specifically stated in the claims that follow. The invention also relates to the corresponding system (simulator), to the objects included in the same, as well as to a corresponding computer program product loadable in the memory of at least one electronic computer and comprising portions of software code to perform the method according to the invention when the product is executed on a computer: in this context such term must be deemed entirely equivalent to means readable by a computer comprising instructions to check a network of computers in order to perform the method according to the invention. The reference to "at least one electronic computer" is obviously aimed to show the possibility of performing the solution according to the invention in a de-centralized context.

In its currently preferred embodiment, the invention solves the above-mentioned problems by introducing for the modules or devices in the simulator respective interfacing objects with the other modules. Such interfacing objects have an "external" side and an "internal" side with respect to module or device. Such external side has an independent character from the

idiosyncrasies of said module or device and is therefore uniform for all system modules or devices.

The term "uniform" obviously means such a character that the "external" side of said interfacing objects is independent from the respective module or device, and can thereby be substantially identical for all modules or devices.

The previously-mentioned problems, intrinsically linked to the possible presence of different implementations of network functionality, are thereby solved, also as regards to the possible intervention of the Statistic Manager or an equivalent entity.

In a particularly preferred way, the following interfacing objects are introduced:

- communication interfaces among all simulated modules and/or devices: interfaces of this type manage the communication with "events" and with "messages" and make implementations of single modules independent from the type of communication that every module is able to perform with other modules in the simulator. Particularly, for every module/device there is a unique communication interface for all different implementations of the considered module; every communication operation among modules and/or devices occurs through the communication interfaces;

- statistic interfaces of simulated modules and/or devices: the interfaces of this type make statistic surveys performed by the Statistic Manager entity independent from the implementation of simulated modules from which statistic data are detected. Particularly, there is an interfacing object for every single implementation of the modules of interest; in this interfacing object, the measuring mechanism of statistic data from the corresponding module is realised in a simple way. The Statistic Manager entity

collects the detected data through the external side, made uniform, of interfacing objects and more directly from single modules.

Brief description of the attached drawings

The invention will be described, purely as a non-limiting example, with reference to the attached drawings, in which:

- figure 1 is a functional approximate diagram of a simulator of the herein described type,
- figure 2 shows the architecture of related communication interfaces,
- figures 3, 4 and 5 show a possible organization of messages and events within a simulator of the herein described type,
- figure 6 is a functional approximate diagram of a statistics managing module in a simulator of the herein described type,
- figures 7 and 8 show in detail the architecture of the relative statistic interfaces,
- figures 9 and 10 are exemplified flow diagrams of the operating modes of the herein described simulator, and
- figure 11 synthetically shows the statistics operation within a simulator of the herein described type.

Detailed description of an embodiment of the invention

Figure 1 shows the architecture of a simulator 10 comprising an engine 11 in which there are all typical managing functionalities of the simulation of a telecommunications network, such as a radio-mobile network, namely:

- Parameter Manager 11a,
- Event Scheduler 11b,
- Factory Manager 11c, and
- Statistic Manager 11d.

A device package 12 is then present in which the different devices 13 are contained, representative of the physical network devices and the objects related to the scenario to simulate.

Every device contains different modules related to the different functionalities managed by the device itself.

Such a simulator, working in general among a set of input signals I and a set of output signals O, can be implemented, for instance, on a computer with Intel Pentium III processor and Microsoft Windows operating system, using Microsoft Visual Studio 6.0 development environment and ANSI C++ programming language.

As shown in figure 2, every module 13', 13'' present in the device package 12 can have many implementations 13b', 13c' or 13b'', 13c'', 13d'', each one related for instance to a set of simulated functionalities or to a specific technology.

The herein described arrangement has the chance of having, for all implementations of a module, a single communication interface 13a', 13a'',... that is used in the information exchange with other modules.

The concept is general and can be applied to any module or device in a cellular network simulator.

Every module has its own communication interface: interface 13a' for module 13' and interface 13a'' for module 13''.

Module 13' has two different implementations 13b' and 13c', and both use interface 13a' to communicate with other modules, as for instance with module 13''.

Module 13'' has three different implementations 13b'', 13c'' and 13d'', and all use interface 13a'' to communicate with the other modules, as for instance with module 13'.

The use of interfaces in the communication among modules allows making the insertion of new implementations of a module simpler and quicker without having to manage from time to time the functionalities related to information exchange.

Every communication interface (or "interfacing object") introduces in fact an "external" side A and an "internal" side B with respect to module or device 13b', 13c', or 13b'', 13c'' and 13d''.

The internal side B reflects the characteristics (or "idiosyncrasies") of the specific module or device and, more particularly, of the specific implementation considered every time.

Contrarily, the external side A of every interfacing object 13a', 13a'' is uniform for all system modules or devices, and therefore for all possible implementations of the same.

The term "uniform" obviously means such character for which the external side A of the interfacing objects is independent from the respective module or device, and can thereby be substantially identical for all modules or devices.

Particularly, the external side A of the interfacing objects is configured in order to allow the information interchange between modules or devices with two modes:

- i) through "messages": the information exchange occurs through the use of objects called "messages".

Depending on the structure shown in figure 3, every "Message" 100 is characterised by an indicator of the source or sender module or device S, by an indicator of the target module or device "Target" 110 and by the information exchanged among modules "Data" 120. The communication with messages is characterized in that the receipt of information from the target

module occurs simultaneously with the transmission from the source module.

- ii) through "events": the information exchange occurs through the use of objects called "events". Depending on the structure shown in figure 4, every "Event" 200 is characterised by an indicator of the source or sender module or device S, by a "Time" 210, by an indicator of the target module or device "Target" 220 and by information exchanged among modules "Data" 230. The communication with events is characterized in that the receipt of information from the target module does not occur simultaneously with the transmission from the source module, but in a particular time instant following the transmission time.

The use of events serves for "delaying" the receipt of an information from the receiving module. The communication with events occurs through the entity called Event Scheduler that can be found in the simulator engine; function of the Event Scheduler is storing events and relative time in a queue, called "queue of events", and sending every event to its own target module when the simulation time reaches the event time itself.

The implementation of every communication interface provides for four main functionalities, you respectively called:

- MessageDispatcher <MessageType>: dispatching functionality of messages of the MessageType type, where MessageType represents any type of message;
- EventDispatcher <EventType>: dispatching functionality of events of the EventType type, where EventType represents any type of event;
- MessageListener <MessageType>: receiving functionality of messages of the MessageType type, where MessageType represents any type of message; and

- EventListener <EventType>: receiving functionality of events of the EventType type, where EventType represents any type of event.

Every dispatching or receiving functionality of messages/events can be found or not in the communication interfaces.

It is possible to have interfaces with only a subset of possible functionalities, for instance an interface can be only MessageListener of a message without necessarily having the other three functionalities.

Besides, in a single interface, the same functionality can be present many times, in order to manage different messages/events. For example, an interface that has to be able to receive two types of different events has two EventListener functionalities: one for the first type of event and another for the second type of event.

As further example (not to be meant as limiting of the scope of the invention) what is shown in figure 5 can be taken into account.

Here there are types of events "Event A" 1000, "Event B" 1010 and "Event C" 1020, the types of messages "Message 1" 2000 and "Message 2" 2010 and the interface related to module "Module ABC" 3000 that it is able of:

- receiving and sending events of the "Event A" type
- receiving events of the "Event B" type
- sending events of the "Event C" type
- receiving message "Message 1", and
- sending message "Message 2"

The related interface has the following functionalities active:

- EventListener <Event A> 3010

- EventDispatcher <Event A> 3020
- EventListener <Event B> 3030
- EventDispatcher <Event C> 3040
- MessageListener <Message 1> 3050, and
- MessageDispatcher <Message 2> 3060

The realization of the simulator interfaces can advantageously be performed with C++ programming language.

In case of the above-described example we have the following code:

```
class Event
{
GenericEventDispatcher * source
GenericEventListener * target
Int timej
}
class EventA
: public Event
{
}

class EventB
: public Event
{
}

class EventC
: public Event
{
}

class Message
{
GenericMessageDispatcher * source
```

```
GenericMessageListener * target
}

class Message1
: public Message
{
}

class Message2
: public Message
{
}

class ModuloABC_Interface
: public EventListener <EventA>
, public EventDispatcher <EventA>
, public EventListener <EventB>
, public EventDispatcher <EventC>
, public MessageListener <Message1>
, public MessageDispatcher <Message2>
{
}

class ModuloABC
: public ModuloABC_Interface
[...]
{
[...] ABC module implementation
}
```

The use of simulator interfaces also involves the architecture related to Statistic Manager.

In the simulator, statistic processing is the task of the Statistic Manager 11d entity that can be found in the engine (see figures 1 and 6).

The Statistic Manager 11d contains the objects that store the various statistics, called "Statistic Calculators".

Figure 6 shows as an example three entities SC1, SC2 and SC3, every one operating as statistic calculator related to the processing of a particular statistic quantity: SC1 can for instance be related to system capacity, SC2 related to mean transmission throughput of the packets, SC3 related to the level of power used in transmission by the simulated radio-mobile terminal.

The operation of the affected computers is independent from the implementations of the simulated modules and is only function of the types of statistic samples to analyse.

The herein described arrangement provides that, for every implementation of a module, an adequate statistic interface is realized, referenced by the name "Data Collector", whose aim is collecting the statistic samples of interest from the corresponding implementation and sending them to the Statistic Manager entity, that proceeds then to their processing.

As a non-limiting example, figure 7 can be taken into account, where there is a device 120 in which two modules 121 and 123 are present.

Module 121 has two different implementations 121a and 122a.

For every implementation, there is a statistic interface (Data Collector): implementation 121a corresponds to statistic interface 121b, implementation 122a corresponds to statistic interface 122b.

Module 123 has three different implementations 123a, 124a and 125a.

For every implementation there is a statistic interface (Data Collector): implementation 123a

corresponds to statistic interface 123b, implementation 124a corresponds to statistic interface 124b, implementation 125a corresponds to statistic interface 125b.

Also in this case, every communication interface or interfacing object has in fact an external side A and an internal side B with respect to module or device 121a, 122a or 123a, 124a and 125a.

The internal side B reflects the characteristics or idiosyncrasies of the specific module or device and, more particularly, of the specific implementation considered every time.

Contrarily, the external side A of every interfacing object 121b, 122b or 123b, 124b, 125b is uniform for all system modules or devices and therefore for all possible implementations of the same.

The external side A of the interfacing objects is therefore independent from the respective module or device, and can thereby be substantially identical for all modules or devices.

As an explanation and example, it can be supposed that module 121 manages the transmission of a packet; in this case a statistic sample of interest is the time employed to transmit every single packet. Therefore, every statistic interface 121b and 122b of the different implementations of module 121a and 121b must be able to identify and collect the time used to transmit a packet from the corresponding implementation of module 121.

In general, the two mentioned implementations of module 121a and 122a totally manage the transmission of a packet in different way, therefore the two corresponding statistic interfaces 121b and 122b have to measure the time of transmission of a packet with a different mechanism.

The implementation of every statistic interface preferably provides for a functionality DataDispatcher <DataType>, namely a dispatching functionality of statistic data of the DataType type, where DataType represents any statistic data type.

As further non-limiting example, what is shown in figure 8 can be taken into account, where in the left part there are the types of statistic data "Data A" 1100, "Data B" 1101 and "Data C" 1102, and in the right part there are the statistic interfaces related to module "Module 1" 1300 and to module "Module 2" 1301.

The interface related to module "Module 1" 1300 are able to send statistic data of the "Data A" 1100 type and of the "Data C" 1102 type.

The interface related to module "Module 2" 1301 are able to send statistic data of the "Data B" 1100 type and of the "Data C" 1102 type.

Then the interface related to module "Module 1" 1300 has the following functionalities active:

- DataDispatcher <Data A> 1300a
- DataDispatcher <Data C> 1300b.

The interface related to module "Module 2" 301 has the following functionalities active:

- DataDispatcher <Data B> 1301a
- DataDispatcher <Data C> 1301b

The realization of statistic interfaces in the simulator is preferably performed with programming language C++.

In case of the above-described example, we have the following code:

```
class DataA
{
[...]
}
```

```
class DataB
{
[...]
}

class DataC
{
[...]
}

class Modulo1_StatisticInterface
: public DataDispatcher <DataA>
, public DataDispatcher <DataC>
{
}

class Modulo2_StatisticInterface
: public DataDispatcher <DataB>
, public DataDispatcher <DataC>
{
}
```

The operation of dispatching a message is shown in figure 9 and is described as follows:

- step 1001: the implementation of sender module sends the message to its own interface MessageType;
- step 1002: the interface of sender module sends the message MessageType to the interface of the receiving module;
- step 1003: the interface of the receiving module receives the message MessageType and sends it to the implementation of the module.

The operation of dispatching an event is shown instead in figure 10 and is described as follows:

- step 2001: the implementation of the sender module sends the event to its own interface EventType, pointing out the time of receipt t;

- step 2002: the interface of the sender module sends Event Scheduler to the entity, the event EventType pointing out the time of receipt t;

- step 2003: when the simulation reaches the time t, the entity Event Scheduler sends the event EventType to the interface of the target module;

- step 2004: the interface of the receiving module receives the event EventType and sends it to the implementation of the module.

The general operation of the method for collecting statistic samples in a given simulator is shown in figure 11, where as an example the Statistic Manager 31 manages three statistics (Statistic Calculator) 31a, 31b and 31c. Statistic 31a processes samples type K, that obtains from a first module having two implementations 32a and 33a, which respectively correspond to statistic interfaces 32b and 33b.

The collection of statistic samples occurs as described as follows:

- during simulation, statistic interfaces 32b and 33b identify samples type K, each one with a proper mechanism that takes into account the functionalities of the corresponding implementations 32a and 33a;

- at every simulation step, the Statistic Manager queries the statistic interfaces so that they send directly collected samples type K to statistic 31a; and

- with a suitable interval, statistic 31a processes samples type K and sends forth the result.

The herein described arrangement essentially involves two main advantages.

Firstly, the communication mode between simulated modules and devices is independent from individual

implementations of modules/devices. In this way, the information exchange modes between modules and devices are established once and for all defining the communication interfaces of every module and device. Every time it is decided to introduce in the simulator a new implementation for a device or for a module, it is not necessary to worry about the management of information exchange because this is already managed.

Secondly, it is possible to rationalize the planning of statistics managed by the Statistic Manager entity since it does not depend from which implementations of devices or modules are used in the simulation. In this way, every time it is decided to introduce in the simulator a new implementation for a device or for a module, it is enough to realize the corresponding statistic interface (Data Collector), without having to modify the operation of the Statistic Calculator.

As a result of the described situation, a flexible and slender general architecture of the simulator is obtained.

As already said, the implementation of the described simulator can be realized with any type of computer, such as Intel, SUN, Apple,... and with any operating system (Windows, Linux, Unix, MAC OS...). The use of ANSI C++ programming language is only a possible choice; the implementation of the simulator can be in fact also performed in other programming languages, such as Java, Delphi, Visual Basic,...

The choice of ANSI C++ language appears currently preferred in view of the good planning flexibility offered by said programming language and of the high level of obtainable performances in the finished program in terms of execution speed.

The invention can be used in cellular networks simulators that simulate other systems besides the mentioned GSM, GPRS and UMTS. The invention can be used in cellular networks simulators that adopt the events simulation technique or the Montecarlo simulation technique.

The skilled technicians in the art will immediately appreciate that the invention does not necessarily deal with the only simulation of radio-mobile cellular networks: the invention can in fact be also used in other types of simulators, where there is a similar architecture with modules and devices complying with real physical equipment and where it is necessary to communicate among various modules/devices the parameters related to simulated functionalities.

It is therefore evident that, notwithstanding the principle of the invention, the realisation particulars and the embodiments can be broadly changed with respect to what has been described and shown, without departing from the scope of the present invention, as defined by the attached claims.